Configuración técnica de Portal y microservicios

Control de versiones

Versión	Fecha de modificación	Responsable	Aprobador	Resumen de cambios
1.0	27/12/2022	Anjana Producto	Anjana Producto	Creación del documento

Configuración Backend

General para todos los microservicios de Anjana

- Los entornos Anjana normalmente usan alias como hecateserver, edusaserver, …

  Por lo que es necesario añadirlos en /etc/hosts utilizado

```
127.0.0.1 indexservice zkservice s3service s3service_node1 rdbservice ldapservice
127.0.0.1 hecateserver hecate1server hecate2server edusaserver zeusserver hermesserver
viatorserver viatorserver_node1 minervaserver kernoserver totserver portunoserver
127.0.0.1 totpluginawsglueserver totpluginawsiamserver totpluginawss3server
totpluginazureadserver totpluginazurefilesserver totpluginazurestorageserver
totplugingcpbigqueryserver totplugingcpiamserver
127.0.0.1 totplugingcpstorageserver totpluginhdfsserver totpluginhiveserver
totpluginjdbcserver totpluginjdbcdenodoserver totpluginjdbcoracleserver
totpluginjdbcredshiftserver totpluginjdbcsqlserverserver totpluginldapserver
totpluginpowerbiserver
```

Pero pueden ser añadidos directamente los nombres de los host, o del servicio en Kubernetes, y no editar /etc/hosts

```yaml
eureka:
  client:
    serviceUrl:
      defaultZone: http://back1.anjanadata.com:50761/eureka
```

- Todos los microservicios de Anjana tienen por defecto un tamaño de de cabeceras HTTP de 2KB, este valor puede ser modificado cambiando esta propiedad en el fichero de propiedades

```yaml
server:
  max-http-header-size: 20000
```

- La url de validación de token en cada microservicio http://<zeus_hostname>:<zeus_port>/internal/v1/auth/validate-token

```yaml
security:
  oauth2:
    resource:
      tokenInfoUri: http://zeusserver:8088/internal/v1/auth/validate-token
```

- Configuración de Hecate Server en cada microservicio

  En Kubernetes es el servicio o el balanceador de carga si hay más de un nodo.

```yaml
eureka:
  client:
    serviceUrl:
      defaultZone: http://hecateserver:50761/eureka
```

- Si se quiere activar las estadísticas de Hibernate con la distinta información que ofrece (Puede afectar al rendimiento por lo que debe estar activa solamente en contadas ocasiones)

```
Jpa:
  properties:
    hibernate:
      generate_statistics: true
```

- A partir de la versión 4.4 si queremos lanzar el servicio para que obtenga la configuración de Edusa habrá que hacerlo con el siguiente comando:

--spring.profiles.active=default
--spring.config.import=configserver:http://<edusa_server>:<port>
--spring.cloud.config.failFast=true

- Todos los módulos de Anjana incluyen un fichero de properties en formato YAML donde se configuran, entre otras propiedades, algunos parámetros de la aplicación. Anjana provee los microservicios con una configuración por defecto. Esa configuración se puede sobreescribir con otros valores. Estas modificaciones podrán ser cargadas si el YAML se encuentra dentro de los ficheros de Edusa o si en los argumentos de arranque de la aplicación se añade:

```
–spring.config.additional-location=<ruta al fichero de propiedades>
```

- Para Hermes, Kerno, Minerva, Portuno y Zeus se puede configurar una lista de url que puede utilizar Swagger para hacer llamadas REST a la API en su yml. En caso de no necesitar Swagger, es una configuración opcional y, por tanto, se puede obviar. Estas urls deberán apuntar al proxy de viator para el correcto funcionamiento de Swagger, si no se incluye, swagger genera una url válida por sí mismo.

```
swagger:
 server:
   url:
     - https://viatorserver/gateway
     - https://<anjana_host>/gateway
     - https://hostalias/gateway
```

- Si se quiere configurar claves (o propiedades de cualquier tipo) de manera que no estén en texto plano, si no encriptadas; se puede hacer colocando la propiedad encriptada en Base64 entre comillas simples y concatenando delante {cipher}. De la siguiente manera:

```
propiedad: '{cipher}asdf8976sdfa'
```

- Colector de basura: Todos los servicios que estén utilizando el jdk8 tendrán que tener el colector de basura G1 para manejar de forma más eficiente la memoria:

```
-XX:+UseG1GC -XX:+UseStringDeduplication
```

Configuración de logs

Los microservicios están configurados por defecto para que den su salida a consola estándar para que los log puedan ser gestionados por la utilidad de logs del sistema, normalmente en entornos de máquina virtual se usará syslog, rsyslog y jornalctl para consumirmos o configurar su tratamiento.

Todos ellos pueden ser configurados mediante el fichero yaml de cada uno siguiendo las prácticas estándar de Spring Boot.

Ejemplo de usos comunes:

```
logging:

  pattern:

      console: "%clr(%d{yyyy-MM-dd HH:mm:ss.SSS}){faint} [HERMES] %clr(${LOG_LEVEL_PATTERN:%5p})
%clr(${PID:-  }){magenta} %clr(---){faint} %clr([%15.15t]){faint} %clr(%-40.40logger{39}){cyan}
%clr(:){faint} %m%n${LOG_EXCEPTION_CONVERSION_WORD:%wEx}"

  level:

    root: INFO

    com.anjana: INFO

    com.netflix.discovery: INFO

    org.hibernate: INFO

    org.activiti: INFO

    #    stat -> if generate_statistics is active show every SQL with performance information

    org.hibernate.stat: DEBUG

       #      StatisticalLoggingSessionEventListener -> if generate_statistics is active shows
metrics after each session

    org.hibernate.engine.internal.StatisticalLoggingSessionEventListener: INFO

                                #                      shows       queries       base       on
spring.jpa.properties.hibernate.session.events.log.LOG_QUERIES_SLOWER_THAN_MS property
```

```
    org.hibernate.SQL_SLOW: INFO

    #   shows query parameters

#    org.hibernate.type.descriptor.sql.BasicBinder: trace

#    lot of information regarding with queries

#    org.hibernate.type: trace
```

Edusa

Edusa es un servicio de configuración central, se puede configurar para servir configuraciones desde ficheros locales o desde un repositorio git.

Tiene que ser lanzado con el argumento —spring.config.additional-location=<path a directorio de fichero de propiedades> para seleccionar el repositorio de propiedades (Filesystem o Git). Es importante que termine en "/" cuando se trata de un directorio, procediendo a leer los ficheros de configuración existentes.

Un ejemplo sería --spring.config.additional-location=file:/opt/data/edusa/conf/

Repositorio Filesystem

- Solo es necesario ajustar la carpeta donde se encuentran los ficheros "/opt/data/…"

Ejemplo:

```
spring:
  profiles: native
  cloud:
    config:
      server:
        native:
         search-locations:
         - file:/opt/data/configrepo
         - file:/opt/data/configrepo/{application}
         - file:/opt/data/configrepo/{application}/{profile}
```

Repositorio Git

Se puede indicar que rama se va a usar en default-label

Ejemplo:

```
spring:
  cloud:
    config:
      server:
        git:
          uri: git@bitbucket.org:anjanadatacom/config-local.git
          default-label: develop
          skipSslValidation: true
          timeout: 10
          clone-on-start: true
          force-pull: true
          searchPaths: '{application}'
          ignoreLocalSshSettings: true
```

```
        privateKey: |
          -----BEGIN RSA PRIVATE KEY-----
                        .....
          -----END RSA PRIVATE KEY-----
```

HA config

Para configurar HA en Edusa, se tiene que modificar el comando de arranque de java en cada microservicio de Anjana añadiendo en el parámetro spring.uri todas las URLs de los servidores donde Edusa está escuchando.

```
[Unit]
Description=Anjana hecate server
Requires=network.target
After=network.target  edusa.service

[Service]
Environment=HOSTNAME=hecateserver
Environment=HECATE_REPLICAS=http://hecateserver:50761/eureka
WorkingDirectory=/opt/hecate
ExecStart=java -Xmx256m -javaagent:/opt/common/xjar-agent-hibernate.jar -jar /opt/hecate/hecate.jar
--spring.cloud.config.uri=http://back1.anjanadata.com:8888,http://back2.anjanadata.com:8888 --spring
.profiles.active=default --spring.cloud.config.failFast=true
User=anjana
Type=simple
Restart=on-failure
RestartSec=10

[Install]
WantedBy=multi-user.target
```

Hecate

- ${HECATE_REPLICAS} en el descriptor de servicio o directamente en el fichero de configuración. Puede ser él mismo, otro nodo o un array de hecateservers
    - http://hecateserver:50761/eureka
    - http://hecateserver1:50761/eureka, http://hecateserver2:50762/eureka

Ejemplo (no necesario si se setea el valor HECATE_REPLICAS) :

```
eureka:
  client:
    serviceUrl:
      defaultZone: http://hecateserver:50761/eureka
```

HA config

Para configurar HA en Hecate, se tiene que modificar los microservicios de Anjana que se registrar en Hecate(Kerno, Minerva, Hermes, Viator, Portuno, Tot, Drittesta, Zeus), cambiando el valor de eureka.client.serviceUrl.defaultZone. En el descriptor de servicio de Hecate, se tiene que cambiar el valor de la variable HECATE_REPLICAS, como se describe en el apartado anterior.

Ejemplo:

```
eureka:
  client:
    serviceUrl:
      defaultZone: http://back1.anjanadata.com:50761/eureka,http://back2.anjanadata.com:50761/eureka
```

Lo mismo pero configurado en el descriptor de servicio:

```
[Unit]
Description=Anjana hecate server
Requires=network.target
After=network.target  edusa.service

[Service]
Environment=HOSTNAME=hecateserver
Environment=HECATE_REPLICAS=http://back1.anjanadata.com:50761/eureka,http://back2.anjanadata.com:50761/eureka
WorkingDirectory=/opt/hecate
ExecStart=java -Xmx256m -javaagent:/opt/common/xjar-agent-hibernate.jar -jar /opt/hecate/hecate.jar --spring.cloud
=http://back1.anjanadata.com:8888,http://back2.anjanadata.com:8888 --spring.profiles.active=default --spring.cloud
lFast=true
User=anjana
Type=simple
```

Viator

- Rutas de accesibilidad a swagger
    - springdoc.api-docs.path: /swagger/v3/api-docs
    - springdoc.swagger-ui.path: /swagger/swagger-ui.html

Kerno

- spring.datasource.url: jdbc:postgresql://<database_host>:<port>/<database>?currentSchema=<schema_kerno>
- spring.datasource.username: el usuario de la conexión a base de datos
- spring.datasource.password: la contraseña de la conexión a base de datos
- spring.jpa.properties.hibernate.default_schema: el mismo schema que en la url
- Spring.jpa.properties.hibernate.generate_statistics: Cuando está a true se generará las estadísticas de cada sesión de hibernate. Luego configurar la variable de log correspondiente al nivel adecuado (*org.hibernate.engine.internal.StatisticalLoggingSessionEventListener*)
- anjana.scheduling: configuración de tareas programadas.
- anjana.scheduling.objectsPerPacket: cantidad de objetos que se incluyen en un paquete a la hora de realizar la indexación masiva.
- anjana.scheduling.lineageObjectsPerPacket: cantidad de objetos que se incluyen en un paquete a la hora de realizar el linaje completo. Para un entorno muy grande (millón y medio de objetos) con muchas relaciones establecer el límite en 25 ó 30. (por defecto = 1000)
- anjana.scheduling.threadScheduleSeconds: tiempo de espera entre que un paquete se prepara y se envía a Minerva para su indexación.
- anjana.scheduling.threads: número de hilos que se usan simultáneamente para el envío de los objetos a Minerva a la hora de realizar la indexación masiva.

- anjana.scheduling.datasource.url: conexión JDBC a portuno que gestiona los bloqueos entre tareas programadas.
- anjana.scheduling.datasource.user: usuario de conexión JDBC a portuno que gestiona los bloqueos entre tareas programadas.
- anjana.scheduling.datasource.password: contraseña de conexión JDBC a portuno que gestiona los bloqueos entre tareas programadas.
- anjana.async.db: configuración del ejecutor de tareas asíncronas relacionadas con las operaciones con base de datos.
- anjana.async.db.maxPoolSize: tamaño máximo de hilos procesando en paralelo cuando la cola de procesos supera el máximo. (por defecto = 20)
- anjana.async.db.corePoolSize: tamaño de hilos procesando en paralelo. (por defecto = 5)
- anjana.async.db.queueCapacity: tamaño de la cola de procesos. (por defecto = 100)
- anjana.async.hermes: configuración del ejecutor de tareas asíncronas relacionadas con las operaciones con hermes.
- anjana.async.hermes.maxPoolSize: tamaño máximo de hilos procesando en paralelo cuando la cola de procesos supera el máximo. (por defecto = 10)
- anjana.async.hermes.corePoolSize: tamaño de hilos procesando en paralelo. (por defecto = 5)
- anjana.async.hermes.queueCapacity: tamaño de la cola de procesos.(por defecto = 1000)
- anjana.async.minerva: configuración del ejecutor de tareas asíncronas relacionadas con las operaciones con minerva.
- anjana.async.minerva.maxPoolSize: tamaño máximo de hilos procesando en paralelo cuando la cola de procesos supera el máximo. (por defecto = 5)
- anjana.async.minerva.corePoolSize: tamaño de hilos procesando en paralelo. (por defecto = 5)
- anjana.async.minerva.queueCapacity: tamaño de la cola de procesos. (por defecto = 100)
- aws.proxy: proxy de Minio. http://<s3_service>:<port>
- aws.idAccount: identificador de la cuenta en Minio.
- aws.accessKey: clave de acceso en Minio.
- aws.secretKey: clave secreta en Minio.
- aws.region: región en la que se encuentra el servidor de Minio
- aws.bucket.imports: nombre de la carpeta necesaria para guardar los ficheros importados desde anjana. Tiene que ser creado. (por defecto = imports)
- aws.bucket.attach: nombre de la carpeta necesaria para guardar los contratos de los DSAs y ficheros asociados a objetos. Tiene que ser creado. (por defecto = dsa)
- Swagger

Minerva

- spring.datasource.url: jdbc:postgresql://<database_host>:<port>/<database>?currentSchema=<schema_kerno>
- spring.datasource.username: el usuario de la conexión a base de datos
- spring.datasource.password: la contraseña de la conexión a base de datos
- spring.jpa.properties.hibernate.default_schema: el mismo schema que en la url
- solr.url: el listado de urls de nodos con instancias de Apache SolR. En caso de más de una instancia separar las urls con “,”.

- solr.client.type: indica el tipo de Apache SolR utilizado. Acepta CLOUD para cluster y HTTP para una única instancia.
- solr.user: el usuario de conexión, en caso de que la seguridad esté habilitada.
- solr.password: la contraseña de conexión, en caso de que la seguridad esté habilitada.
- solr.security: flag que indica si la seguridad está habilitada en los nodos de Apache SolR.
- Swagger

Zeus

- spring.datasource.url: jdbc:postgresql://<database_host>:<port>/<database>?currentSchema=<schema_zeus>
- spring.datasource.username: el usuario de la conexión a base de datos
- spring.datasource.password: la contraseña de la conexión a base de datos
- spring.jpa.properties.hibernate.defaul_schema: el mismo schema que en la url
- eureka.client.serviceUrl.defaultZone : Url de eureka.
- anjana.group.crossName: nombre utilizado en el perfil de usuario para agrupar los roles transversales (*cross*) a la organización.
- anjana.role.defaultRole: nombre del rol que se le asigna por defecto a todos los usuarios, estén incluidos en alguna unidad organizativa o no. Si no se incluye no se tendrá en cuenta.
- anjana.scheduling: configuración de tareas programadas.
- anjana.scheduling.datasource.url: conexión JDBC a portuno que gestiona los bloqueos entre tareas programadas.
- anjana.scheduling.datasource.user: usuario de conexión JDBC a portuno que gestiona los bloqueos entre tareas programadas.
- anjana.scheduling.datasource.password: contraseña de conexión JDBC a portuno que gestiona los bloqueos entre tareas programadas.
- anjana.scheduling.licenseNotification : Batch para la comprobación de la licencia.
- anjana.scheduling.notification : Batch para el envío de notificaciones para informar al administrador de los usuarios que no están marcados como cross en los roles.
- anjana.bcrypt.strength : Longitud del algoritmo bcrypt que se usa para encriptar las contraseñas cuando la autenticación y autorización es por base de datos.
- anjana.mail.azure : Nombre de los atributos donde se recoge el mail para azure. Por defecto es en los campos otherMails, mail y userPrincipalName.
- anjana.mail.aws : Nombre de los atributos donde se recoge el mail para aws. Por defecto es en el campo email.
- anjana.mail.google : Nombre de los atributos donde se recoge el mail para google. Por defecto primaryEmail y emails.
- anjana.mail.db : Nombre de los atributos donde se recoge el mail para base de datos. Por defecto email.
- anjana.mail.ldap : Nombre de los atributos donde se recoge el mail para ldap. Por defecto mail.
- eureka.internal.kerno.name : Nombre del microservicio kerno en eureka.
- eureka.internal.drittesta.name : Nombre del microservicio drittesta en eureka.

Security tiene dos partes (security.authentication y security.authorization) :

- security.authentication.ldap.user-authentication: autenticación del usuario sobre LDAP. USER_PASSWORD, USER_PASSWORD_ENC y USER_CONNECTION son los valores permitidos.
- security.authentication.ldap.url: URL y puerto de LDAP.
- security.authentication.ldap.base-dn: DN base del esquema de LDAP.
- security.authentication.ldap.user-search-filter: filtro para la búsqueda de usuarios por el atributo seleccionado en LDAP.
- security.authentication.ldap.user-structural-class: clase estructural de LDAP para los usuarios.
- security.authentication.ldap.user-search-attribute: atributo para la búsqueda de usuarios en LDAP.
- security.authentication.ldap.connection-user-dn: DN del usuario administrador de LDAP que se va a usar para establecer la conexión.
- security.authentication.ldap.connection-user-password:contraseña del usuario administrador de LDAP.
- security.authentication.db.enable: flag para activar/desactivar la autenticación por base de datos.
- security.session.expiredtimeminutes: tiempo de expiración de la sesión en minutos.
- security.jwt.expiration-timeout-minutes: tiempo de expiración del token en minutos.
- security.jwt.secret-key: clave para codificación del token.
- security.authentication.oidc.providers: agrupación de providers a los que Zeus tendrá acceso.
- security.authentication.oidc.providers.<name>: clave para darle nombre a cada una de las conexiones.
- security.authentication.oidc.providers.<name>.name: nombre para mostrar en el Portal y Portuno.
- security.authentication.oidc.providers.<name>.authorize-url: URL del proveedor para autorizar usuarios de Anjana Portal. (Utiliza variables en la URL).
- security.authentication.oidc.providers.<name>.authorize-url-portuno: URL del proveedor para autorizar usuarios de Portuno. (Utiliza variables en la URL).
- security.authentication.oidc.providers.<name>.token-url: URL propia del proveedor para gestionar la creación de tokens.
- security.authentication.oidc.providers.<name>.scopes: scopes de autenticación propios del proveedor.
- security.authentication.oidc.providers.<name>.client-id: identificador de autenticación en el proveedor.
- security.authentication.oidc.providers.<name>.client-secret: secreto de autenticación en el proveedor.
- security.authentication.oidc.providers.<name>.client-authentication-method: método de autenticación propio del proveedor.
- security.authentication.oidc.providers.<name>.redirect-uri: URI a la que tiene que redirigir el navegador cuando se produce un login exitoso en un proveedor externo de autenticación en el portal.
- security.authentication.oidc.providers.<name>.redirect-uri-portuno: URI a la que tiene que redirigir el navegador cuando se produce un login exitoso en un proveedor externo de autenticación en el portal administrativo (Portuno).
- security.authentication.oidc.providers.<name>.username-claim: campo donde se encuentra el nombre de usuario en el proveedor.
- security.authentication.oidc.providers.<name>.type: tipo de proveedor.
- spring.authentication.oidc.providers.{proveedor}.workflowType: IMPLICIT

- Con valor IMPLICIT funcionará todo como antes
- Con valor AUTHORIZATION_CODE seguirá este flujo definido por OAuth2
  - En caso de poner este valor se necesitan configurar dos propiedades más a la misma altura de workflowType:
    - authorizeServer: url del endpoint del servidor de autorización que devuelve información del usuario en base a un 'authorization code'
      - AWS: `https://<your-user-pool-domain>/oauth2/userInfo` Doc AWS
      - GCP: Doc GCP
      - Azure: https://graph.microsoft.com/oidc/userinfo Doc Azure
      - Auth0: https://<auth-server>/userinfo Doc Auth0
    - userNameProperty: propiedad donde se encuentra el nombre del usuario en el endpoint anteriormente mencionado. Suele coincidir con la propiedad usernameClaim ya configurada para el proveedor.
      - AWS: username
      - GCP: email
      - Azure: email
      - Auth0:email

- security.authorization.ldap.url: URL y puerto de LDAP.
- security.authorization.ldap.base-dn: DN base del esquema de LDAP.
- security.authorization.ldap.connection-user-dn: DN del usuario administrador de LDAP que se va a usar para establecer la conexión.
- security.authorization.ldap.connection-user-password:contraseña del usuario administrador de LDAP.
- security.authorization.ldap.user-structural-class: clase estructural de LDAP para los usuarios.
- security.authorization.ldap.user-search-attribute: atributo para la búsqueda de usuarios en LDAP.
- security.authorization.ldap.group-structural-class: clase estructural de LDAP para los grupos.
- security.authorization.ldap.group-search-attribute: atributo para la búsqueda de grupos en LDAP.
- security.authorization.ldap.member-of-patch-filter: filtro para especificar la membresía de un usuario en un grupo.
- security.authorization.ldap.membership-user-group: atributo de pertenencia de un usuario en un grupo.
- security.authorization.ldap.root-ous: unidades organizativas raíz cuyo nombre se ha de excluir en la composición del nombre de la lista de unidades organizativas en su recuperación.
- security.authorization.ldap.ou-structural-class: clase estructural de LDAP para las unidades organizativas.
- security.authorization.ldap.ou-search-attribute: atributo para la búsqueda de unidades organizativas en LDAP.
- security.authorization.db.enable: flag para activar/desactivar la autorización por base de datos.
- security.authorization.{providerType}.providers.{providerName}.groupOrgUnitSeparator: propiedad que permite configurar el separador de unidades organizativas.
- security.authorization.{providerType}.providers.{providerName}.roleOrgUnitSeparator: propiedad que permite configurar el separador de roles.

- security.authorization.{providerType}.providers.{providerName}.groupPrefix: propiedad que permite poder configurar el borrado de un prefijo al obtener las unidades organizativas en en el proovedor. Si no se incluye, por defecto no borrará nada.
- Swagger

Hermes

- anjana.activiti.data: flag que especifica si se van a cargar los flujos en el arranque.
- anjana.activiti.output: ruta temporal de ficheros que necesita Activiti.
- anjana.activiti.input: ruta en la que se encuentran los ficheros XML en caso de que se haya especificado la carga en el arranque.
- wf.async.waiter: tiempo de espera para los intentos de sincronización entre Anjana y Activiti. (por defecto = 20s)
- wf.async.retries: cantidad de reintentos de sincronización entre Anjana y Activiti. (por defecto = 3)
- wf.validation.automatic : Indica con true o false si el workflow tendrá validación automática.
- spring.datasource.url: jdbc:postgresql://<database_host>:<port>/<database>?currentSchema=<schema_hermes>
- spring.datasource.username: el usuario de la conexión a base de datos
- spring.datasource.password: la contraseña de la conexión a base de datos
- spring.datasource.hikari.schema: el mismo schema que la url
- spring.jpa.properties.hibernate.defaul_schema: el mismo schema que en la url
- spring.activiti.databaseSchema: el mismo schema que la url
- anjana.scheduling: configuración de tareas programadas.
- spring.minio.url: proxy de Minio. http://<s3_service>:<port>
- spring.minio.accessKey: clave de acceso en Minio.
- spring.minio.secretKey: clave secreta en Minio.
- spring.minio.bucket: nombre de la carpeta necesaria para guardar los xml de los workflows. Tiene que ser creado. (por defecto = workflows)
- spring.mail.host: servidor de STMP
- spring.mail.port: puerto del servidor de STMP
- spring.mail.username: usuario del servidor de STMP
- spring.mail.password: contraseña del servidor STMP
- spring.mail.from: correo remitente por defecto
- spring.mail.properties.mail.auth: activar/desactivar la autorización del servidor STMP
- spring.mail.properties.mail.starttls.enable: activar/desactivar el TTLS
- Swagger

Drittesta

- spring.datasource.url: jdbc:postgresql://<database_host>:<port>/<database>?currentSchema=<schema_hermes>
- spring.datasource.username: el usuario de la conexión a base de datos
- spring.datasource.password: la contraseña de la conexión a base de datos
- spring.jpa.properties.hibernate.defaul_schema: el mismo schema que en la url

- veltesta.host: url de Veltesta para validar la licencia
- veltesta.port: puerto de acceso a Veltesta para validar la licencia
- veltesta.protocol: protocolo de acceso a Veltesta para validar la licencia
- anjana.scheduling: configuración de tareas programadas.
- anjana.scheduling.objectsPerPacket: cantidad de objetos que se incluyen en un paquete a la hora de realizar la indexación masiva.
- anjana.scheduling.threadScheduleSeconds: tiempo de espera entre que un paquete se prepara y se envía a Minerva para su indexación.
- anjana.scheduling.threads: número de hilos que se usan simultáneamente para el envío de los objetos a Minerva a la hora de realizar la indexación masiva.
- anjana.scheduling.datasource.url: conexión JDBC a portuno que gestiona los bloqueos entre tareas programadas.
- anjana.scheduling.datasource.user: usuario de conexión JDBC a portuno que gestiona los bloqueos entre tareas programadas.
- anjana.scheduling.datasource.password: contraseña de conexión JDBC a portuno que gestiona los bloqueos entre tareas programadas.
- anjana.installationCode: código de instalación úinica (Provisionado por Anjana)
- anjana.anjanaPublicKey: clave única y pública de la instalación (Provisionado por Anjana)
- anjana.privateKey: clave única y privada de la instalación. Puede ser provisionada por Anjana o generada por el cliente de la siguiente manera (en este caso el cliente proporcionará su correspondiente pública para la instalación).
- application.services.kerno.name : Nombre del microservicio kerno, esta propiedad sirve para llamar correctamente a kerno si se hubiera cambiado su nombre en eureka.
- application.services.veltesta.name : Nombre del microservicio veltesta, esta propiedad sirve para llamar correctamente a veltesta si se hubiera cambiado su nombre en eureka.

Generación de claves de encriptación

El NIST recomienda usar claves de como mínimo 2048 bits, que son las que proporciona Anjana. Aún así, si el cliente lo desea, podrá generar un nuevo par de claves RSA para incrementar la seguridad de la encriptación hasta 4096 bits creando las claves de forma manual; de este modo el cliente será el creador y validador de la seguridad y unicidad de las claves. Una vez generadas, la pública la deberá proporcionar a Anjana para configurar su instalación. La privada se establecerá en el proyecto de **Drittesta** con la nueva propiedad del yaml **anjana.privateKey** teniendo en cuenta que:

- Representa la clave privada del cliente para la comunicación entre los servicios de licencia.
- Será una propiedad obligatoria.
- Podrá ser generada por el cliente de la siguiente manera:
    - Con el endpoint habilitado en **Drittesta** para ello (clave de 2048 bits):

```
curl --location --request GET 'http://{{host}}:{{port}}/api/license/keys'
```

Donde {{host}} es la dirección de la máquina y {{port}} el puerto de **Drittesta.** La respuesta es un json con el siguiente formato:

```
{
```

```
  "publicKey": clave pública,
  "privateKey": clave privada
}
```

- O bien,

En una consola UNIX ejecutar las instrucciones:

- ssh-keygen -m PKCS8 -t rsa -b 4096
- openssl rsa -in nombreClave -pubout

La primera genera las claves en un directorio local. Descripción de los parámetros:

- **-m** indica el tipo de algoritmo que es obligatorio que sea PKCS8
- **-t** indica el tipo de clave que es RSA obligatoriamente.
- **-b** indica el tamaño de la clave. El cliente puede configurar el tamaño que desee. Los tamaños más comunes son 1024, 2048, 3072 o 4096 bits, siendo 4096 el tamaño máximo.

  La segunda instrucción devuelve la clave pública en el formato correcto para usarse en Anjana. El parámetro "nombreClave" será el nombre de la clave que hemos dado en el paso anterior.

Portuno

- Hay varias conexiones a base de datos, una por cada schema en base de datos: anjana, hermes, minerva, zeus y portuno. Todas tienen el mismo formato:
  - <schema>.datasource.url: jdbc:postgresql://<database_host>:<port>/<database>?currentSchema=<schema>
  - <schema>.datasource.username: el usuario de la conexión a base de datos
  - <schema>.datasource.password: la contraseña de la conexión a base de datos
  - <schema>.datasource.hikari.schema: el mismo schema que la url
  - <schema>.jpa.properties.hibernate.defaul_schema: el mismo schema que en la url
- spring.minio.url: proxy de Minio. http://<s3_service>:<port>
- spring.minio.accessKey: clave de acceso en Minio.
- spring.minio.secretKey: clave secreta en Minio.
- spring.minio.bucket: nombre de la carpeta necesaria para guardar los xml de los workflows. Tiene que ser creado. (por defecto = workflows)
- spring.minio.bucketTranslations: nombre de la carpeta necesaria para guardar los ficheros cdn. Tiene que ser creado. (por defecto = cdn)
- spring.minio.folderTranslations: nombre de la carpeta necesaria para guardar los ficheros de traducción. Tiene que ser creado. (por defecto = i18n)
- application.services.kerno.name: Nombre del microservicio kerno, esta propiedad sirve para llamar correctamente a kerno si se hubiera cambiado su nombre en eureka.
- application.services.hermes.name: Nombre del microservicio hermes, esta propiedad sirve para llamar correctamente a hermes si se hubiera cambiado su nombre en eureka.

- application.services.minerva.name: Nombre del microservicio minerva, esta propiedad sirve para llamar correctamente a minerva si se hubiera cambiado su nombre en eureka.
- application.services.zeus.name: Nombre del microservicio zeus, esta propiedad sirve para llamar correctamente a zeus si se hubiera cambiado su nombre en eureka.
- Swagger

Tot

La comunicación entre Tot y los plugins se realiza de forma cifrada, encriptando el mensaje que se envía y se recibe. Por ello, es necesario informar de cuál es la clave privada de Tot y de la clave pública de los plugins (usarán todos la misma).

- tot.privateKey: Clave privada de Tot
- tot.plugin.publicKey: Clave pública de los plugins
- anjana.async.pool.maxPoolSize: Tamaño máximo del pool de hilos (valor por defecto 3)
- anjana.async.pool.corePoolSize: Tamaño de cores usado (valor por defecto 3)
- anjana.async.pool.queueCapacity: Máximo nº de hilos en cola (valor por defecto 1500)

Tot plugins

En cada plugin hay que añadir la configuración de claves para cifrar la comunicación con Tot.

- totplugin.encryption.privateKey: Clave privada del plugin
- totplugin.encryption.tot.publicKey: Clave pública de Tot

Para generar un nuevo par de claves válido se recomienda usar los siguientes comandos. El primer comando usa el algoritmo RSA con codificación PKCS8 para generar las claves.

```
ssh-keygen -m PKCS8 -t rsa
```

Es importante no introducir ningún texto cuando solicite una passphrase, solo pulsar Enter, para que no devuelva la propia clave encriptada.

El comando anterior generará dos archivos key y key.pub (key siendo el nombre introducido para el fichero de salida). El archivo key.pub contendrá la clave pública y el archivo key la privada. Para obtener la clave pública en el mismo formato que la privada, se debe introducir el siguiente comando, que formateará la clave pública y la dejará lista en el nuevo fichero keyPublic, para usarla en los plugins.

```
openssl rsa -in key2 -pubout -out keyPublic
```

Además de la configuración común, cada plugin tiene un documento para desplegar y configurar.

- Tot plugin Sql Server
- Tot plugin LDAP
- Tot plugin jdbc
- Tot plugin GCP IAM
- Tot plugin GCP BigQuery
- Tot plugin GCP Storage
- Tot plugin Azure Storage
- Tot plugin Azure Files
- Tot plugin Azure AD
- Tot plugin AWS S3
- Tot plugin AWS IAM
- Tot plugin PowerBi
- Tot plugin Oracle database

Apache2

HA config

Para configurar HA en Apache, se tiene que modificar las entradas Location en la configuración de apache2, añadiendo las correspondientes IP/dominios de destino para los nodos existentes.

Por ejemplo, para añadir nodos de minio, se necesita cambiar el siguiente ProxyPass.

```
<Proxy balancer://cdnbackends>
      BalancerMember http://s3service_node1:9000/cdn
      BalancerMember http://s3service_node2:9000/cdn

      ProxySet lbmethod=bytraffic
</Proxy>
<Location /cdn>
      ProxyPass balancer://cdnbackends
</Location>

<Proxy balancer://dsabackends>
      BalancerMember http://s3service_node1:9000/dsa
      BalancerMember http://s3service_node2:9000/dsa

      ProxySet lbmethod=bytraffic
</Proxy>
<Location /dsa>
      ProxyPass balancer://dsabackends
</Location>

<Proxy balancer://importsbackends>
      BalancerMember http://s3service_node1:9000/imports
      BalancerMember http://s3service_node2:9000/imports

      ProxySet lbmethod=bytraffic
</Proxy>
<Location /imports>
      ProxyPass balancer://importsbackends
</Location>

<Proxy balancer://miniobackends>
      BalancerMember http://s3service_node1:9000/
      BalancerMember http://s3service_node2:9000/
      ProxySet lbmethod=bytraffic
</Proxy>
<Location /minio>
      ProxyPass balancer://miniobackends
</Location>
```

Lo mismo para Viator

```
<Proxy balancer://gatewaybackends>
      BalancerMember http://viatorserver_node1:8085
      BalancerMember http://viatorserver_node2:8085

      ProxySet lbmethod=bytraffic
</Proxy>
<Location /gateway>
      ProxyPass balancer://gatewaybackends
</Location>
```

Igual para Portuno

```
<Proxy balancer://portunobackends>
      BalancerMember http://portunoserver1:8998/portuno
      BalancerMember http://portunoserver2:8998/portuno

      ProxySet lbmethod=bytraffic
</Proxy>
<Location /portuno>
      ProxyPass balancer://portunobackends
</Location>
```

Se ha añadido la posibilidad de disponibilizar los endpoints y parámetros de todos los microservicios a través del swagger accediendo por viator, para hacer uso de ello habría que descomentar la siguiente entrada:

```
#  UNCOMMENT TO USE SWAGGER URL <anjana_url>/swagger/swagger-ui.html
#  <Proxy balancer://swaggerbackends>
#        BalancerMember http://viatorserver_node1:8085/swagger timeout=5
#        ProxySet lbmethod=bytraffic
#  </Proxy>
#  <Location "/swagger">
#        ProxyPass "balancer://swaggerbackends"
```

```
#       ProxyPassReverse "balancer://swaggerbackends"
#  </Location>

#  <Proxy balancer://swaggerapibackends>
#       BalancerMember http://viatorserver_node1:8085/v3/api-docs timeout=5
#       ProxySet lbmethod=bytraffic
#  </Proxy>
#  <Location "/v3/api-docs">
#       ProxyPass "balancer://swaggerapibackends"
#       ProxyPassReverse "balancer://swaggerapibackends"
#  </Location>
```

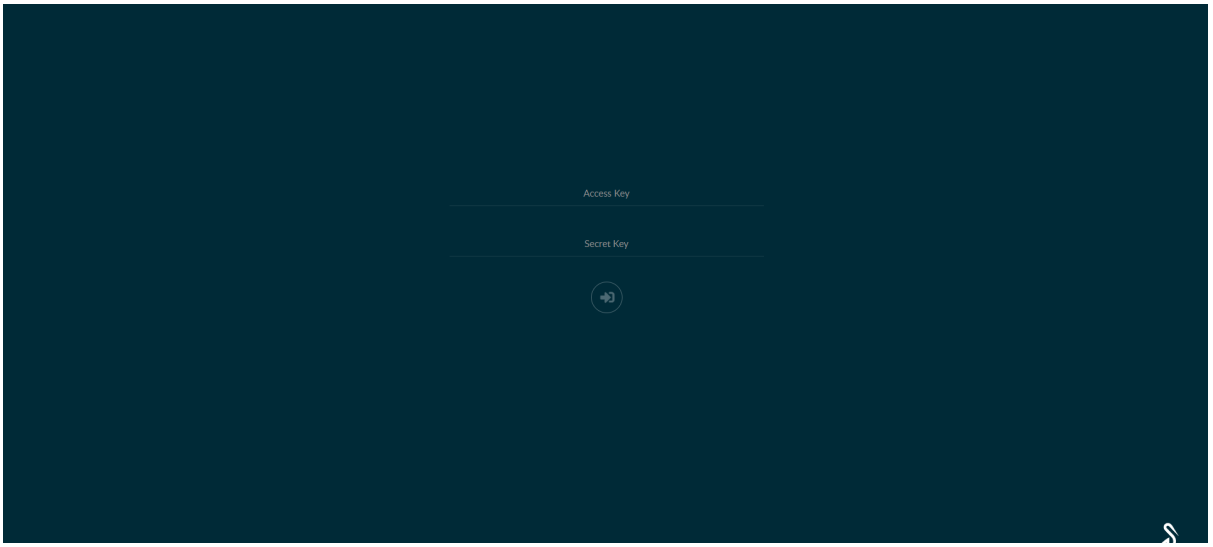
```
#  UNCOMMENT TO USE SWAGGER URL <anjana_url>/swagger/swagger-ui.html
#  <Proxy balancer://swaggerbackends>
#    BalancerMember http://viatorserver_node1:8085/swagger timeout=5
#    ProxySet lbmethod=bytraffic
#  </Proxy>
#  <Location "/swagger">
#    ProxyPass "balancer://swaggerbackends"
#    ProxyPassReverse "balancer://swaggerbackends"
#  </Location>

#  <Proxy balancer://swaggerapibackends>
#    BalancerMember http://viatorserver_node1:8085/v3/api-docs timeout=5
#    ProxySet lbmethod=bytraffic
#  </Proxy>
#  <Location "/v3/api-docs">
#    ProxyPass "balancer://swaggerapibackends"
#    ProxyPassReverse "balancer://swaggerapibackends"
#  </Location>
```

Configuración Frontend

Minio

¿Qué es Minio?

Anjana usa almacenamiento de ficheros en servidores cloud compatible con Amazon S3. Como almacén de datos, Minio permite guardar datos no estructurados como fotos, videos, backups o imágenes de contenedores.



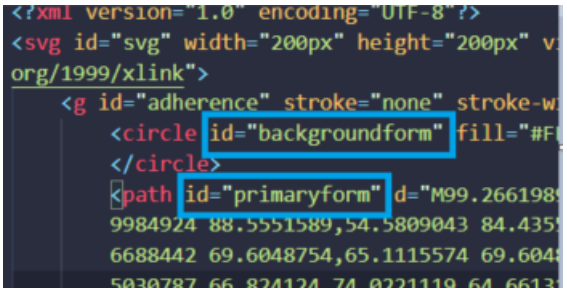
Minio funciona usando contenedores llamados **buckets.**

Estructura de Minio básica en Anjana Data

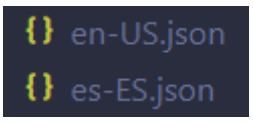
Para el correcto funcionamiento de Anjana es necesario crear los siguientes buckets:

- **imports** -> para almacenar los ficheros excel para importación masiva de objetos
- **dsa** -> para almacenar los contratos de los DSA y cualquiera de los ficheros adjuntos a las entidades o relaciones de Anjana.
- **cdn** -> para almacenar las imágenes de la aplicación y los ficheros de traducción de los idiomas utilizados en la misma.
    - images
        - brand: logotipos de imagen corporativa
        - error-pages: imágenes de páginas de error
        - icons: para almacenar los iconos propios del core de la aplicación (no personalizables).
        - languages-icons: iconos de idiomas
        - profile: imagen de perfil de usuario
        - subtype-icons: para almacenar los iconos de subtipo personalizables. Estos iconos están hechos con una sola forma fundida (primaryform) y tiene un borde que será el que indica el estado del objeto en la pantalla de linaje.

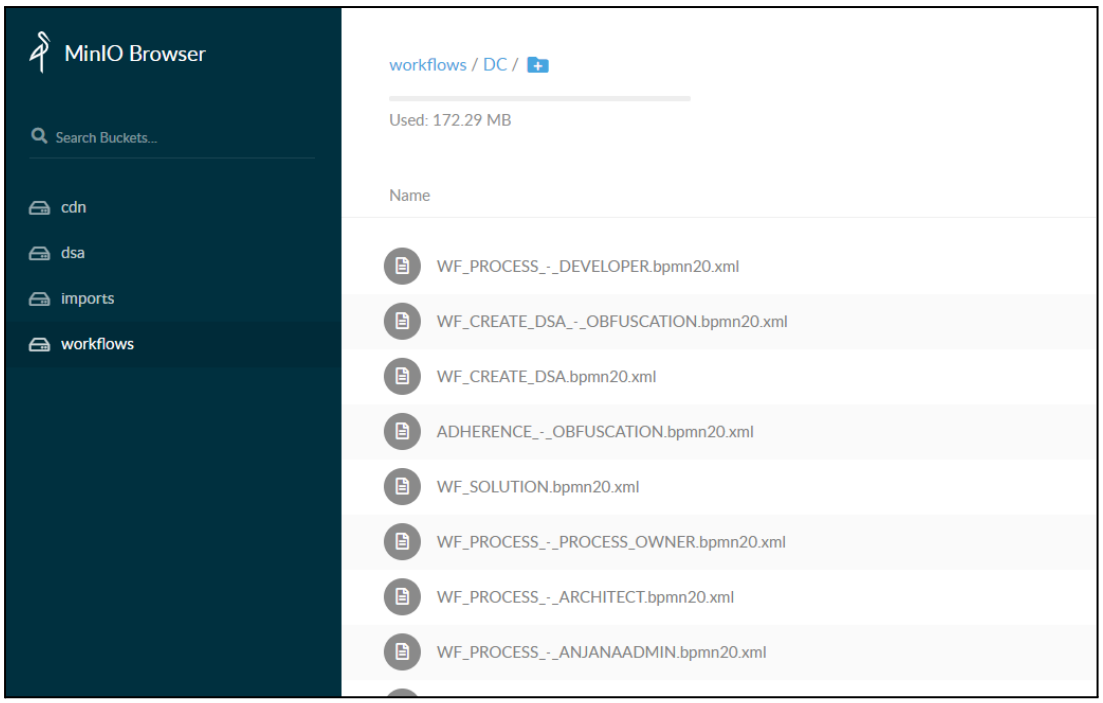
          Para que esto funcione, el círculo de fondo que llevan los iconos de subtipo tiene el id="backgroundForm", Y la forma del icono es un path con el id="primaryForm", sin estos ids no funcionarán los iconos en el linaje.



    - i18n -> donde se guardan los ficheros que se generan, desde Portuno, con las traducciones para permitir en Anjana el multiidioma



- **workflows** -> para almacenar los xml correspondientes a la configuración de los workflows que se van a ejecutar en Anjana. Este bucket contiene dos buckets: BG y DC



CDN - Precarga estática

Tienen que ser precargados los ficheros estáticos en el bucket CDN para que la aplicación funcione correctamente (iconos, imágenes, ficheros de traducción, etc).

```
wget
https://<user>:<pass>@artifactory.anjanadata.org:8443/repository/anjanareleasesraw/com/anjana/cdn/<v
ersion>/cdn-<version>.tar

tar -xzpvf cdn-<version>.tar -C <path de la carpeta cdn>
```

Notas

Desde Anjana v.4.2.7, se usa una nueva versión de Minio, desde la cual la consola ha sido desenlazada de la API y es necesario añadir un vhost en el fichero /opt/apache/conf/httpd.conf para tener acceso.

```
<VirtualHost *:8081>
    ServerAdmin admin@test.com
    ServerName test.com
    ServerAlias www.test.com

  <Proxy balancer://miniobackends>
        BalancerMember http://s3service_node1:9001/
        ProxySet lbmethod=bytraffic
```

```
  </Proxy>
  <Location />
        ProxyPass http://s3service_node1:9001/
  </Location>
</VirtualHost>
```